



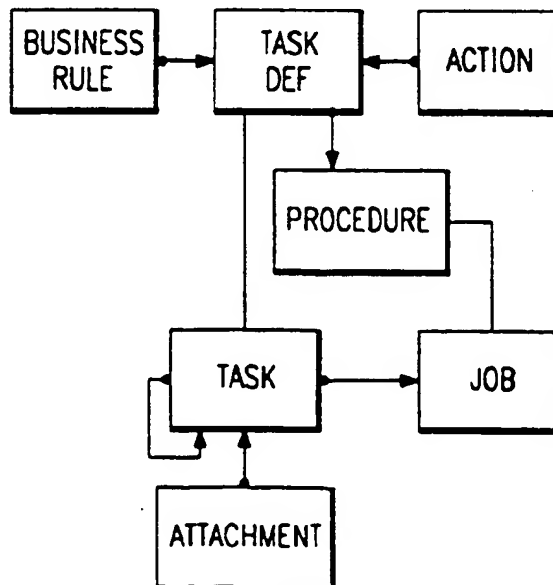
## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification <sup>5</sup> : <b>G06F 15/21</b>		<b>A1</b>	(11) International Publication Number: <b>WO 94/29804</b> (43) International Publication Date: 22 December 1994 (22.12.94)
(21) International Application Number: <b>PCT/US94/06688</b> (22) International Filing Date: 14 June 1994 (14.06.94)  (30) Priority Data: 08/078,663          16 June 1993 (16.06.93)          US  (71) Applicant: <b>ELECTRONIC DATA SYSTEMS CORPORATION [US/US]; 5400 Legacy Drive, M/S H-3A-05, Plano, TX 75024 (US).</b>  (72) Inventor: <b>KOKO, Boma, Richard; 10111 Margo Lane, Westminster, CA 92683 (US).</b>  (74) Agent: <b>GRIEBENOW, L., Joy; Electronic Data Systems Corporation, 5400 Legacy Drive, M/S H3-3A-05, Plano, TX 75024 (US).</b>			(81) Designated States: <b>AU, BR, CA, JP, RU, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).</b>  <b>Published</b> <i>With international search report.          Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i>

(54) Title: **PROCESS MANAGEMENT SYSTEM**

## (57) Abstract

A computer-based method of modeling and managing processes followed by a business enterprise. A process is modeled as a series of tasks. During a task disposition process, each task may have a number of different states. Tasks change state in response to performance of actions, which can be accomplished manually or by automated means. Business rules represent conditions that determine whether an action can be performed. Business rule handlers may be invoked to determine whether business rule conditions are met. Action handlers may be invoked to determine how an action is to be executed.



**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgyzstan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

## PROCESS MANAGEMENT SYSTEM

TECHNICAL FIELD OF THE INVENTION

5        This invention relates to computer-aided business management, and more particularly to a method for managing the processes followed by a business enterprise.

BACKGROUND OF THE INVENTION

"Product data management" (PDM) is a term used to describe computer-based methods for managing product design and manufacture. An example of a PDM system is the Information Manager system, sold by Electronic Data Systems. The design of the Information Manager system is based on the objects it manipulates. The primary focus of the system is on representing the enterprise in terms of its objects and operations on them. Object classes are derived by modeling enterprise operations such as design, manufacture, administration, project management, and cost control.

Apart from PDM systems, various other computer-based management systems have been developed to assist in business process management. One such product is the Power Frame system, a product of Digital Equipment Corporation. Another such product is the KI Shell system, a product of Universal Energy Systems. However, neither of these systems provides or is easily integrated with PDM tasks.

Another disadvantage of existing computer based-decision making systems is the heavy data entry burden on the business enterprise that is to implement the system. The enterprise must model each job and the business rules that apply to it.

SUMMARY OF THE INVENTION

5 A method of using a computer to manage a process followed by a business enterprise. During a process definition phase of operation, the computer receives process definition data that specifies the process to be followed as a series of tasks, and stores the process definition data as a model of the process. The computer system stores a set of actions that may be performed on tasks, to change said tasks from a current state to a next state. It also stores a set, of business rules that determine whether an action can be performed, each business rule having at least one condition. During a job management phase of operation, the computer receives job specification data that specifies a job to be performed in accordance with the process. It then assembles a task disposition list representing at least one task to be performed during the job and the state of said task. It outputs the task disposition list to a user, and receives action input from a user representing an action to be taken toward disposition of the task. It evaluates a business rule associated with the task, and performs the action if the business rule is satisfied. It then changes the state of said task to a next state. This process is repeated for each task, until all tasks of a job have been completed.

25 An advantage of the EPM system is that it provides a means by which a business enterprise can describe its processes. The system can then create instances of those processes in terms of "jobs" that it manages. It applies business rules to these processes so that consistent results are achieved. It manages data produced during a job and associates the data to the job.

35 In general, EPM 12 ensures that processes are properly executed so that the right tasks are performed at the right time by the right person. It notifies assignees of tasks that their task is ready for action and provides that person with relevant data produced during previously

performed tasks. The system is especially useful for training new members of an enterprise who perform certain processes as described to the EPM system by experts.

5 Another advantage of the EPM system is that it provides a means for a business enterprise to model its processes "generically". At the same time, it permits the attachment of data to a particular job on an "ad hoc" basis. This combination of generic process models with the ability to attach job-specific data provides the enterprise  
10 with the ability to deal with change and job-to-job differences, while encouraging consistency and repeatability to the extent practicable.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a computer-based PDM system, with which an EPM in accordance with the invention is used.

5 Figure 2 illustrates the basic steps of a computer-based method of managing a process in accordance with the invention.

Figure 3 illustrates the data classes used by EPM 12 to model a process.

10 Figure 4 is an example of a process definition dialog.

Figure 5 is an example of a task definition dialog.

Figure 6 is an example of a handler definition dialog.

Figure 7 is a syntax diagram of the main body of a task definition as stored by EPM.

Figure 8 is a syntax diagram of a business rule.

15 Figure 9 is a syntax diagram of an action handler or business rule handler.

Figure 10 is an example of a job status dialog.

Figure 11 is an example of task disposition dialog.

Figure 12 is an example of a user's in-box.

20 Figure 13 illustrates a change request form for initiating a job.

Figure 14 illustrates an in-box for the change request job.

25 Figure 15 illustrates a task disposition dialog for the change request job.

Figure 16 illustrates a disposition of a task of the change request job.

Figure 17 illustrates a list of tasks and other objects associated with the change request job.

## DETAILED DESCRIPTION OF THE INVENTION

### System Overview

Figure 1 illustrates a computer system for implementing a product data manager (PDM) 10, with which an enterprise process manager (EPM) 12 is integrated. EPM 12 is a type of PDM module that deals with modeling and managing processes in accordance with the invention described herein. As stated in the background section, an example of a PDM system 10, without EPM 12, is the Information Manager, a product of Electronic Data Systems.

PDM 10 is stored in memory of, and is executed by, a conventional computer system 11, such as a VAX/VMS or a UNIX system. Typically, the computer system is part of a distributed network of workstations having a number of computers 11. In the example of this description, the operating system includes a windows type sub-system, which supports various graphical user interfaces, such as dialog boxes and selection buttons. Computer 11 is in communication with input and output devices, which for purposes of this description are a keyboard, pointing device, and graphics display.

EPM 12 may be integrated with other PDM modules 12a, which implement various PDM tasks. An advantage of implementing EPM 12 as a part of a more comprehensive set of PDM modules 12a is that it can then make use of data from other program modules and deliver data to them. For example, a product structure manager (PSM) module might model the design of a product to be manufactured, with data from that module being provided to EPM 12 to indicate that certain components are to be assembled. An enterprise resource manager (ERM) module might model how resources such as materials and employees are made available for use by the processes modeled by the EPM 12.

As explained below, EPM 12 stores a model of at least one process, as specified by a user. Specification of a

process is based on the assumption that most activities of an enterprise are sufficiently structured to be described with a given syntax. In this connection, process steps are modeled as tasks. Actions performed as part of a task may  
5 be affected by business rules associated with those actions.

The computer programming used to implement EPM 12 is based on object-oriented design. Thus, data is associated with classes, which have hierarchies and relationships.  
10 Classes specify what data they store and what operations can be performed on them. Instances of data classes are objects, and are derived by modeling the operations of various application domains. It is representations of these objects that are manipulated by user interface 16,  
15 with graphical icons, dialogs, and selection buttons. As explained below, tasks, business rules, and actions are data classes whose objects represent a particular enterprise.

EPM 12 also stores "handlers", which are run-time functions defined for actions or business rules.  
20 Evaluation of a business rule or execution of an action may invoke an associated handler, which may be as simple as some sort of flag checking function or may be as complex as a decision support program. These handlers provide for  
25 process management that is customized to a particular enterprise. More specifically, actions and business rules can be defined generally, with different handlers being used to determine how actions are performed and how business rules are evaluated.

Although EPM 12 permits customized actions, business rules, and handlers, an advantage of the invention is that many aspects of business enterprises can be modeled "generically". That is, many enterprises share common business rules. Also, many actions involved in performance  
30 of tasks are common to all tasks, such as "assign" or "start" or "complete". Likewise, handlers for business  
35

rules and actions may be common to various enterprises. Database 13 stores these generic objects and functions, so that a particular business enterprise need only describe those objects that are unique to it.

5 PDM platform 14 provides a base upon which the rest of the PDM system 10 is built. It has several modules, including a persistent object manager (POM). The POM provides the following services: mapping object representation to relational representation, messaging, and  
10 concurrent access control. In general, platform layer 14 isolates PSM 12 and other PDM modules 12a from the operating system and other sub-systems of computer 11.

User interface layer 16 is comprised of user application programming built on the underlying  
15 architecture. Because EPM 12 and other PDM modules 12a are designed for customization via user interface 16, they comply with the programming strategy often referred to as "toolkit" design.

Figure 2 illustrates the basic steps of a computer-  
20 based method of managing a process performed by a business enterprise, in accordance with the invention. Figure 2 is an overview; each step is discussed in detail in connection with Figures 3 - 12.

A characteristic of the method is that it is  
25 interactive with a user. Thus, some steps of the method involve receiving input from a user. Also, different types of users within an enterprise might be involved during different steps. With respect to steps 202 - 208, a system administrator might be responsible for entering business  
30 rules and actions and process and task definitions. With respect to step 212, a supervisor might be responsible for creating jobs. With respect to step 218, a worker to whom a task is assigned might be responsible for entering actions. However, the method is the same regardless  
35 whether a single user, or different users, take these roles. It should also be understood, that in some cases,

the "user" might be other programming that serves PDM 12 by providing data input. For purposes of this description, these various types of users are referred to as simply "the user".

5           Operation of PDM 12 has two phases. A first phase is a process definition phase, in which business rules, actions, and tasks associated with a process are defined and stored in memory. Steps 202-210 comprise this process definition phase. A second phase is a process execution  
10           phase, in which specific instances of the process are managed, on a task by task basis, to disposition. Steps 212-229 comprise this second phase. In a more general sense, these stages are set-up and run-time phases, respectively.

15           Step 202 is storing a set of business rules to be followed during the process. Each business rule is an "if-then" type rule, having a set of conditions associated with its "if" side and at least one "action" associated with its "then" side. As explained below, the action is performed  
20           only if the conditions are met. Business rules state what should be done or what should not be done in specific situations. They may manifest themselves as constraints or as collections of empirical knowledge. Some rules govern other rules, by stating when other rules can be ignored or  
25           when other rules must be applied. A business rule does not change data, but rather, determines when and if an action can be performed.

          Step 204 is storing a set of actions that may be performed to change the state of any task. In general,  
30           actions are activities whose successful completion cause a change of state of a task. The following are examples of actions: assign, start, do, complete, suspend, resume, skip, abort, refuse, and undo. Examples of task states are: unassigned, pending, started, completed, suspended,  
35           skipped, and aborted. Appendix A is a table of the relationships between states and actions. An exception to

the general case of an action resulting in a change of state is the action, do, which can be repeatedly executed without a change of state.

5 Step 206 is receiving data representing a process definition. A "process" is defined as an ordered sequence of tasks performed by an enterprise in order to accomplish the purpose of its business. Step 208 is receiving data from a user representing definitions of tasks of the process. These definitions may include references to  
10 action handlers and to business rule handlers, whose programming is stored in accordance with steps 202 and 204. The tasks may call for either manual or automated disposition. For example, a manual task might require a physical prototype of a new product to be built, whereas an  
15 automated task might require a geometric model to be modified.

Step 210 is storing the process definition data and the task definition data as a process model, to which specific instances can be mapped during run-time process  
20 management.

Step 212 is receiving a job specification. A job is a specific instance of a process, and represents a process to be managed by the run-time aspects of the invention.

Step 214 is assembling a task list for the job  
25 specified in step 212. This list may be sorted and combined with other task lists, such as to list tasks assigned to a particular person. The list may be formatted into various displays of tasks to be performed together with other relevant data, such task states and the person  
30 to whom each task has been assigned.

Step 216 is presenting a task to a user, whether in the form of an in-box or some other form. Step 218 is receiving data, representing explicitly or implicitly, some action to be performed for the task. In this sense, EPM 12  
35 "prods" the user for task disposition and waits for input.

Step 220 is evaluating any business rule associated with the action requested for that task. This includes execution of any business rule handlers specified for that action in the task. The business rule evaluation returns  
5 a "go" or "no go" decision to indicate whether its conditions permit a proposed action to be performed, i.e., whether the business rule is satisfied.

Step 222 is reached only if step 220 returns a "go" decision. In step 222, the action is performed, which may  
10 include execution of any action handlers specified for that task and that action.

Step 224 is performed if the action returns no errors, and results in a change of state of the task.

#### 15 Process and Task Definition

Figure 3 illustrates the data classes used by EPM 12 to model a process with data that represents tasks, business rules, and actions. As described above, task objects are defined by specifying business rules and  
20 actions associated with them. Business rule conditions govern actions, in the sense that no action can be performed if its associated conditions are not met. A task is related to other tasks by the job specification in which it is included. Objects of other data classes may also be  
25 attached to a task, and may be carried from a previous task to a successor task, so that relevant data is retained.

Appendix B describes several of the data classes used by EPM 12 in further detail. Consistent with object-oriented data representation, these data classes have  
30 methods and attributes associated with them.

Because actions and business rules are not always instantaneous, the data associated with a task and the state of the task are persistent. In the case of business rules, the persistent result of business rule evaluation is  
35 stored by its corresponding task.

Figures 4 - 6 are examples of how a human user enters a process definition and task definitions during the process definition phase of operation. This data may include references to other data already stored, such as to actions, business rules, and handlers. The entered data and the stored data is used by EPM 12 to create the objects of Figure 3 and assemble a process model.

Figure 4 is a process definition dialog 40, with which the user enters data to define a process followed by a particular enterprise. The user specifies a unique process name, and optionally, a process description. The user also lists tasks of the process. Each process has a root task that takes the name of the process. All other tasks are added as sub-tasks of the root task or as sub-tasks of other sub-tasks.

A task may be a declaration that another already-defined process should be started. Such a task is indicated by a "Y" in the "invoked" column of the process definition dialog.

Figure 5 is a task definition dialog 50, with which a user enters data to define tasks. The user enters a task name, and optionally, a task description. Tasks are "re-useable" in the sense that task names are not unique to a process and different processes may include the same task.

For each action available for a task, the user may specify one or more action handlers. For example, in Figure 5, the action, start, has two handlers. One handler inherits data from the previous task. The other handler sets protection for objects during this task. Thus, even though different processes may include the same task, the task can be performed differently during different jobs, by invoking a different handler. Appendix C describes the inherit handler as well as other examples of action handlers.

For each business rule associated with an action, the user specifies one or more business rule handlers. In the

example of Figure 5, there are two business rule handlers, check-role and check-completion. Appendix C describes the check-completion handler as well as other examples of business rule handlers.

5           A business rule can be defined as a number of sub-rules, each with its own handler. If this is the case, the user may also specify a quorum number, which is less than the number of business rules. This permits EPM 12 to consider a business rule as satisfied if a quorum of  
10 handlers for that rule return a "go" decision. The user may also specify an override privilege for a business rule handler, as indicated in the "priv" column, to permit one handler to override others. The user may also specify that a business rule handler can be negated, as indicated in the  
15 "not" column. In general, the specification of a quorum, an override privilege, or negation, are alternative means of satisfying a business rule associated with a task.

          An example of a typical business rule is a rule that protects an object created during a job once it has reached  
20 certain stages of development. For example, if the job is one of product development, business rules may allow it to be changed only by certain members of an enterprise.

          Figure 6 is a handler definition dialog 60 for declaring an action handler or a business rule handler for  
25 a particular action of a task. The user enters a name, and optionally, one or more arguments.

          After a process and its tasks have been defined, any business rules or handlers that are not already stored in database 13 must be loaded. Appendix B sets out examples  
30 of several action handlers and business rule handlers. The business rule handlers correspond to "generic" business rules that any business might be expected to follow. Likewise, the action handlers correspond to how certain actions may be carried out in a manner common to a number  
35 of different enterprises. Thus, the handlers of Appendix

B are examples of "generic" handlers that might be stored in libraries of database 13.

Some business rules, actions, and handlers might not be "generic" , in that they are unique to a particular enterprise, but may be re-useable for different processes, jobs, or tasks. This re-usability of handlers promotes consistency within an enterprise.

Using the data entered by the user, EPM 12 assembles a process model. The following code form is illustrative of a process model stored by EPM 12:

```

PROCESS ::= PROCESS TASK-DEF-BODY

TASK-DEF ::= TASK-DEF-BODY

TASK-DEF-BODY ::= name [description]
                { [business-rule-definition]+
                  [action-definition]+
                  [{TASK-DEF TASK-DECL}]+
                }

TASK-DECL ::= PROCESS process-name ;

business-rule-definition ::= BUSINESS-RULE action-id
[<n>]
{
    [business-rule-handler-definition]+
}

business-rule-handler-definition ::= name ( [string]+
[NOT] [OVERRIDE]

handler ::= ACTION action-id name ( [string]+ )

action-id ::= {INITIALIZE | ASSIGN | START | DO |
35 COMPLETE | SUSPEND | RESUME | SKIP | ABORT | UNDO |
REJECT}

```

Figures 7 - 9 are syntax diagrams of the process model. Specifically, Figure 7 illustrates the syntax of a task, as a recursive set of the task definitions for a process. The keyword, process, starts a process definition if followed by a task-body. A process is a special case of a task, i.e., a root task, and is associated with a process definition file.

The keyword, process, may also appear in the definition of a task, in which case it represents an already defined process. The keyword, task, starts the definition of a task. It is followed by a task-body, which  
 5 contains zero or more actions, business rules, or other task definitions and declarations.

The keyword, action, represents an action that may be performed. It may or may not invoke a user-specified handler. Regardless of whether there is an action handler,  
 10 the keyword, business rule, invokes a business rule handler to determine whether conditions permit a proposed action to be allowed.

Figure 8 illustrates the syntax of a business rule, for determining whether an action, identified as A-ID, should be allowed. The model includes a handler and any  
 15 override or negation data such as specified with dialogs 50 and 60.

Figure 9 illustrates the syntax of a handler. A handler has a string of argument data, which are passed to the handler during job management. A tag to identify the  
 20 job and task is also passed.

EPM 12 allows for process flow control that involves task branching. More specifically, business rules can be defined to model OR conditions by ensuring that only a  
 25 desired task has its business rule(s) satisfied for a particular case. OR conditions are exemplified by what-if, if-then-else, and case-of conditions. An example of a case-of model, where Task A breaks to peer Tasks B - D, which are followed by Task E, is:

```

30 TASK A {
    TASK E {
        TASK B {
            BUSINESS-RULE START 2 /* quorum = 2 */
            {
35             EPM-task-state( C::started ) NOT
                EPM-task-state( D::started ) NOT
                MANUAL-OK()  OVERRIDE /* User does
                    this. */
            }
        }
    }
}

```

```

    }
TASK C
    {
        BUSINESS-RULE START 2 /* quorum = 2 */
5          {
            EPM-task-state( B::started ) NOT
            EPM-task-state( D::started ) NOT
            MANUAL-OK() OVERRIDE /* User does
10          this. */
          }
    }

TASK D {
15    BUSINESS-RULE START 2 /* quorum = 2 */
        {
            EPM-task-state( C::started ) NOT
            EPM-task-state( B::started ) NOT
            MANUAL-OK() OVERRIDE /* User does
20          this. */
        }
    }
}

```

25 An AND condition is one in which several tasks must be completed before a subsequent task can be performed. This is modeled by making the tasks to be AND'd arguments of a check-completion rule in a subsequent task. The following is an example of an AND model, where Task E cannot be

30 started until Tasks B - D have been completed:

```

TASK E {
    BUSINESS-RULE START{
        CHECK-COMPLETION (B C D)
35    }
}

```

#### Process Management

Once a process and its tasks have been defined, handlers have been stored, and a process model assembled,

40 EPM 12 may manage specific instances of that process, which are referred to herein as "jobs". In this aspect of its operation, EPM 12 permits users to create jobs and dispose of tasks.

45 Figure 10 is an example of a job dialog box 100, such as a user might use to create a new job or review the

status of its tasks. Here, the user is a supervisor, Mary Ann, who has entered a job named ECO-125-A. EPM 12 automatically lists the tasks for this job.

5       When a job is created, each of its tasks has an unassigned state. However, by using the "assign" button of dialog 100, a user can assign a task. In the example of Figure 10, the "responsibility" column indicates that tasks have already been assigned. Instead of a single person, the user might enter data representing a resource pool, and  
10       an assignment automatically made by EPM 12 to a member of that pool. In either case, the "assign" action might invoke one or more action handlers or business rules that govern, respectively, how the assignment is to be carried out, or whether the requested assignment is permissible.

15       The job is in progress, as indicated by the differing task states. Once a job is in progress, its tasks are always each in a known state.

      The buttons of the dialog box 100 permit the user to see or edit details of each task. The user may select a task, and then a button to "open" the task.  
20       

      The job dialog 100 of Figure 10 is an example of a means for a user to explicitly create a job. An example of how a user might implicitly cause EPM 12 to create a job is discussed below in connection with Figure 13.

25       Figure 11 is an example of a task disposition dialog 110. As an example of when EPM 12 generates dialog 110, a user might use a job dialog 100 to select the task "validate change request" and then the "open" button.

      Each task has an extendable list of objects attached to it. In general, these objects are instances of the  
30       other data classes discussed above in connection with Figure 3. Via dialog 110, the user can add, delete, modify, or simply view objects associated with the task. For example, the user might select the object DRW-125-A,  
35       which is a drawing, and use the "open" button to view the drawing. The ability to edit objects created by other PDM

modules is an advantage of integrating EPM 12 with other PDM modules.

5 By entering data to dialog 110, a user can perform an action and change the state of a task. The user can change the state of the task by selecting an appropriate button. The permitted actions are a matter of presentation, in that buttons for all or only some actions may be displayed.

10 Figure 12 is an example of an "in-box" 120 for reviewing the state of jobs assigned to a particular person or resource pool. An in-box 120 is a means for EPM 12 to inform a user that a task needs action. Referring again to Figure 10, the specification of a job includes assigning responsibility for performance of each task to a particular person. Once the assignment is made, the task appears in that person's in-box 120. In other words, an in-box 120 is a display of tasks assigned to a particular person. An example of when EPM 12 displays in-box 120 is in response to a user's explicit inquiry for all tasks assigned to him or to his resource pool.

20 As with tasks listed in the job description dialog 100, a task from an in-box 120 can be selected and opened. This permits a person, such as the person who is performing the task, to add or delete data attached to the task. A user, depending on his role and privileges, can see the job of which his task is a part, the definition of the job, and conditions that have been or need to be met.

25 The "open job" button of dialog 120 permits the user to view a display of the tasks of the entire job, such as by means of dialog 100. In general, any user may view and edit any data associated with a job, for which he has access rights.

30 Dialog 110 and in-box 120 are two examples of task disposition dialogs, with which a user can explicitly request EPM 12 to perform an action. Dialog 110 lists tasks according to job, and in-box 120 lists tasks

according to user. In each case, selection of a button is an example of a user's explicit change of state of a task.

5 EPM 12 also accommodates a change in state that is implied from a user's input. For example, a task might be for a user to examine some data. When the user's examination is complete, the user might enter data into a computer-generated form to implicitly indicate completion.

10 Regardless of whether the user explicitly or implicitly requests an action to be performed, any associated business rule is are evaluated in accordance with the method of Figure 2. If the business rule is not satisfied or if the action cannot be performed, the state of the task is not changed. Subsequent events, such as a change of conditions, may permit the action to be  
15 successfully performed.

A feature of EPM 12 is that although it relies on user-initiated actions, it attempts to move jobs toward completion. To this end, the performance of certain actions may automatically call certain action handlers.  
20 For example, when a task's start action is requested, all sub-tasks are requested to complete. Depending on the business rules of these sub-tasks, they may or may not actually change state. As another example, if a complete action is requested, all succeeding tasks are requested to  
25 start. Some will start because according to their business rule, all they were waiting for was the task just completed.

#### Example of Process Management

30 Figures 13 - 17 illustrate an example of how EPM 12 is used for run-time process management. The run-time process assumes that a process has been defined, including the business rules and actions associated with its tasks.

35 Figure 13 illustrates a display of a "change request" form 130 generated by EPM 12 or other PDM modules 12a in response to a user's request. In this case, data was

entered to the form by a user named Mary Ann, who identified the job as ECR-125-A and assigned it to Bob. The job involves a change to design of a product component. By pressing an "OK" button, the user invokes a "change request save" handler, which causes EPM 12 to create the job and attach a change request disposition form to the job.

Figure 14 illustrates an in-box 140, which lists one or more of the tasks of the job ECR-125-A. The in-box is that of Mary Ann's and Bob's manager, Jim, who can view the job at the job level or "open" the job to view tasks and objects. Other buttons represent various actions that a user can request.

Figure 15 illustrates a task disposition dialog 150. It was created in response to an "open" button of the in-box 140. This dialog 150 corresponds to the change request disposition form created in response to the change request job creation form 130. Bob, or whomever he might re-assign the task to, may attach other information to the task as necessary. For example, he might enter an object in the subject list box 151, of object type Note, to indicate that the task should not have been assigned to him.

Figure 16 illustrates a disposition of a task of the job, ECR-125-A. A user, Mike, has been assigned the task of executing a change order. He has created the change order, ECO-125-A, and attached a drawing, L-123-E, to be changed.

Figure 17 illustrates a task list 170 of various tasks and other objects of the job, ECR-125-A. The task list 170 indicates that Mike has assigned the task of changing the drawing to Leroy. As a result, Leroy will find this task in his in-box. The task will have inherited the data created in the previous task. Leroy may open the task, which changes its state to "started". He may then select the drawing attached to the task, make the necessary

changes, and request a completion action, which changes the state to "completed".

Other Embodiments

5           Although the invention has been described with  
reference to specific embodiments, this description is not  
meant to be construed in a limiting sense. Various  
modifications of the disclosed embodiments, as well as  
alternative embodiments, will be apparent to persons  
10 skilled in the art. It is, therefore, contemplated that  
the appended claims will cover all modifications that fall  
within the true scope of the invention.

	State	Action	State
	unassigned pending started	assign	pending pending started
5	pending started	start complete	started completed
	unassigned pending started	suspend	suspended
10	suspended	resume	unassigned pending started
	unassigned pending started suspended	skip	completed
15	unassigned pending started suspended	abort	aborted
	pending	refuse	unassigned
20	start unassigned completed pending	undo	pending
25			

**Task****Implementation Class Methods**

5      Currently only a job can create a task.      The create function is not part of the public interface.

- createInstance

10      **Public Instance Methods**

- name()      the name as defined in the script.
- description()
- responsibleParty()
- 15      •      assignResponsibility()
- attachObject(tag\_t, int attachment\_type)

20      Objects are attached to the task for particular reasons or one can say a particular type of relation is created between task and object. This is represented by setting the appropriate type. Current types include:

- 25      •      target      the reason for the task. Fix drawing x. Drawing x will be the target of the task
- reference      reference information quite similar to the current RLM concept.
- 30      •      status      usually a task form will be attached with this type.
- decision      for signoff purposes a signoff is attached as a decision. These are not inherited from task to task.

- 35      •      attachedObjects(int\*,int\* types,tag\_t\*\*) return the attached objects and their attachment types.
- detachObject() detach an object from the task.
- parentProcess() return the job of which this task is a part.

40      •      state() This function returns the state\_value.

- nextTask()
- rootTask()
- previousTask()
- callingTask()
- 45      •      checkRule( const char\* ruleName )      This method will evaluate the specified rule.

- 50      •      performAction(EPM\_task\_action\_t,tag\_t data)

5 This method will test for violation of intrinsic rules  
 such as invalid state change(~~completed-to-started~~).  
 It will then check any associated business  
 preconditions. If all is well it will perform the  
 specified action by executing the declared and  
 registered action. The result of this execution (no  
 errors) will determine whether or not the state value  
 is actually changed. If an action is not declared the  
 state value is simply changed.

10

#### Protected/Private (Implementation) Instance Methods

- promote()
- checkCriteria(int entry/exit)
- 15 Evaluate the business rules associated with the entry  
 criteria. Return {EPM\_go|EPM\_nogo|EPM\_undecided}  
 • state(newState)
- 20 This function sets the state of the task.  
 • initialize( process, node\_id,first\_target\_object)  
 • load() load task from database.  
 • taskDefinition() return task definition.  
 • removeReference(int) remove attachment at given  
 25 position.  
 • executeHandlers(whatToDo, Data)  
 • assign(newParty)  
 • start()  
 • complete()  
 30 • suspend()  
 • resume()  
 • skip()  
 • abort()  
 • refuse()  
 35 • undo()

#### Attributes

- parent\_process tag
- 40 • attachments isA tag\_list
- attachment\_types isA integer list
- responsible\_party tag of user/group/etc....
- state\_value {EPM\_unknown | EPM\_unassigned |  
 EPM\_assigned | EPM\_started | EPM\_completed |  
 45 EPM\_suspended | EPM\_undone |  
 EPM\_aborted | EPM\_skipped } isA integer
- priority\_value

- sub\_process\_id a means of match task as stored in the database with their definition in the script.

5

**Process****Class Methods**

10

- createInstance

**Instance Methods**

15

- traverse

**Attributes**

20

- name
- description string
- task\_handler isA TaskHandler
- entry\_criteria SetOf(TaskHandler)
- exit\_criteria SetOf(TaskHandler)

**EPM File**

25

This class provides facilities to store the definitions of processes persistently. It provides facilities to compile these files and locate them.

30

**Class Methods**

- createInstance
- lookup
- compile

35

**Instance Methods**

- loopUpProcess

40

**Attributes**

- tag\_t

**Handler**

45

50

When a handler is called, it is passed a message that contains the tag to the instance of a task and the type of message (ASSIGN, START, DO, COMPLETE, etc.). For business rules this indicates what we wish the rule to be evaluated for.

A message will also contain a list of arguments for which we shall provide a means of iterating over. Currently macros are being used in the prototype.

- 5     •     void init\_argument\_list(argument\_list)
- char\* next\_argument(argument\_list)
- int number\_of\_arguments(argument\_list)

#### Class Methods

- 10     •     createInstance

#### Instance Methods

- 15     •     apply()

This function will invoke the handler. The handler could be a simple function that displays a dialog asking a user to say what state the task is in.

- 20     A task that takes a while to execute can return "EPM\_started". We shall be by again some time and we shall use the function state() to ask the questions.

#### 25     Attributes

- name
- function\_pointer
- number\_of\_arguments
- 30     •     arguments

#### Job

- 35     This class provides facilities to monitor the run-time process and make reports of the status of the job. The facilities here and in Task may be enhanced to connect to other PDM modules. It inherits most of its behavior from Task.

#### 40     Class Methods

createInstance(processName, jobId, jobDescription, first\_target\_object)

#### 45     InstanceMethods

rootTask()

- 50     task( int sub\_process\_id )     return the task with the matching subprocess id.

**Implementation Instance Methods (Protected/Private)**

```
initialization()
load()
5 selectEntireJob()
loadEntireJob()
deleteEntireJob()
addNewTask()
10 processDefinitionFileName()

Attributes

definition_file      isA string --
definition_name
15 root_task_tag
def_file_tag
name                isA string
```

### Action Handlers

#### ~~auto-assign-rest~~

5        This handler attempts to assign the users/groups listed in  
its arguments to the remaining tasks in the process. If  
none of them satisfy the assignment criteria, the current  
user/group will be tried as a final resort. The purpose is  
10        to allow tasks to be run. Tasks cannot be run until  
resources/responsibility are assigned. It is ideal for use  
as the entry-handler for the root task. If used as such,  
then the creator of a task job can perform as many tasks as  
possible immediately when the job is created.

#### 15        ~~process-signoff~~

The signoff handler maintains information for a particular  
instance of a sign off list. It will however, be extended  
to account for the fact that we have added QUORUM,  
20        OVERRIDE, and DENY to its model.

#### ~~do-task~~

25        This handler provides a means of displaying the generic  
task disposition form and allowing us to record the state  
of a manual task.

#### ~~set-change-level~~

30        This handler sets the change level of an object to the  
specified one in the list of levels established for this  
object type. It will expect to find a file;  
"change-level.definitions" in a data directory. The format  
for this file shall simply be:

35        object-type = "level10" "level1".

The order of the levels is implied by position. When this  
function appears in a process definition, the object-type  
40        and level is provided. If the target folder for that task  
contains only objects of a particular type, the type  
specification may have only the level specified.

45        object-type = "level10" "level1".

**advance-change-level**

5 This handler is a specialization of the set-change-level. It increases the level of each object in the target folder associated with the task by the number of levels specified by its integer argument.

**inherit**

10 This handler provides a means for collecting some or all of the data of a previous task. It takes several arguments. Each argument can specify a type of attachment to inherit. For example, a task can be specified to inherit all target attachments from the previous, parent or root task.

15

**notify**

20 The arguments to this handler are user-ids, group-ids or distribution list ids.

**prompt-user**

25 This handler provides an easy way to display a given form at appropriate times during a job to solicit information from the end user.

**undo-continue**

30 A process at a current task can be undone all the way back to a task that can start a re-work.

**Business Rule Handlers****check-state**

35

This handler will check the state of a given task in the same job. If the job is found to have the specified state it will return a go decision. Otherwise, it will return a nogo decision. If for any reason it cannot determine the state to be what is required it will return a nogo.

40

**check-completion**

45 This handler will take a list of argument strings that specify the name of tasks that must be completed before the one for which its business rule is specified as a constraint can start. EPM will pass these strings to the handler function. It does not interpret the strings. This means that the strings can themselves contain not only the

task that needs to be completed but possibly the expected result.

#### **~~perform-signoff~~**

5

Any time before a decision is made on an entire signoff list associated with a task, the rule will return a "nogo" to EPM. EPM will evaluate this result and those of the other constraints and proceed accordingly.

10

```
SIGNOFF ( "QUORUM=3" "department::role" "individual=user-id[DENY]" "group=group-id" "role")
```

15

The arguments in a signoff handler can be of any of the above forms. It can specify a quorum of voters. It can specify an individual, a particular group, or a particular role. It can specify that any of these signees has override and/or deny privilege.

20

#### **~~allowed-role-list~~**

25

This handler allows users to specify what role a user must have in order to perform a particular action. The rule will ensure that responsibility is assigned only when this criteria is met and the action can only be performed by users with the specified set of roles.

30

The arguments of this rule will be of the form  
[<group\_name>::]{\*|<role\_name>}

#### **~~assigner-role-list~~**

35

This handler ensures that only users whose role satisfies the specified list will be allowed to assign/re-assign the task. In all other respects it is like the Allowed-Role-List.

WHAT IS CLAIMED IS:

1. A method of using a computer to manage a process followed by a business enterprise, comprising the steps of:
  - receiving process definition data from a user,
  - 5 specifying a process to be followed as a series of tasks;
  - storing said process definition data as a model of said process;
  - storing a set of actions that may be performed on said tasks, to change said tasks from a current state to a next
  - 10 state;
  - storing a set of business rules that each determine whether an action can be performed, each business rule having at least one handler for determining whether a business rule condition has been satisfied;
  - 15 receiving job specification data from a user, specifying a job to be performed in accordance with said process;
  - assembling a task disposition list representing at least one task to be performed during said job and the
  - 20 state of said task;
  - displaying at least a portion of said task disposition list to a user;
  - receiving action input representing an action to be taken toward disposition of said task;
  - 25 evaluating a business rule associated with said action, by executing said handler;
  - performing said action if said business rule is satisfied;
  - changing the state of said task; and
  - 30 repeating said displaying, receiving, evaluating, performing, and changing steps for each task of said job.

2. The method of Claim 1, wherein said step of receiving job specification data further comprises receiving assignment data that specifies at least one person by whom a task may be performed.

5

3. The method of Claim 1, further comprising the step of receiving task definition data for at least one task, which specifies at least one action handler, and the step of performing a computer process in accordance with said handler after said step of receiving action input.

10

4. The method of Claim 1, further comprising the step of receiving task definition data, which specifies a business rule associated with at least one action of that task.

15

5. The method of Claim 1, further comprising the step of receiving task definition data for at least one task, which specifies at least one business rule handler, and wherein said step of evaluating a business rule includes performing a computer process in accordance with said handler.

20

6. The method of Claim 5, wherein said steps of specifying at least one business rule handler and said step of evaluating a business rule are performed for more than one handler.

25

7. The method of Claim 1, wherein said action input is generated by said computer.

30

8. The method of Claim 1, wherein action input is in response to manual input.

9. The method of Claim 1, further comprising the step of sorting a number of task list to generate a list of tasks to be performed by a particular user.

5           10. The method of Claim 1, wherein said tasks are represented as data objects, and further comprising the step of associating other data objects to said tasks.

10           11. The method of Claim 1, wherein said step of storing a set of actions includes storing a do action, and wherein said step of performing said action may be repeated before said step of changing the state of said task.

15           12. The method of Claim 1, wherein said step of evaluating a business rule is repeated for a number of tasks such that only one task can have an action performed.

20           13. The method of Claim 1, wherein said step of receiving task definition data comprises specifying a business rule handler such that at least one of its actions cannot be performed until another task has been completed.

25           14. The method of Claim 1, wherein said assembling step comprises assembling a list of tasks to be performed by a specified user.

          15. The method of Claim 1, wherein said assembling step comprises assembling a list of tasks of said job.

30           16. The method of Claim 1, wherein said changing step is followed by the step of automatically changing the state of other tasks.

17. The method of Claim 1, wherein said displaying step comprises displaying each task of said job and its current state.

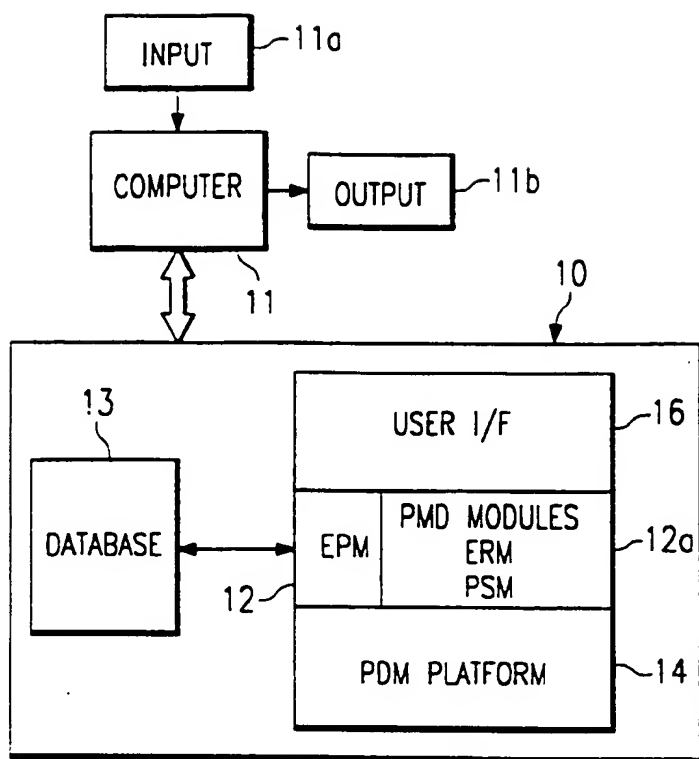


FIG. 1

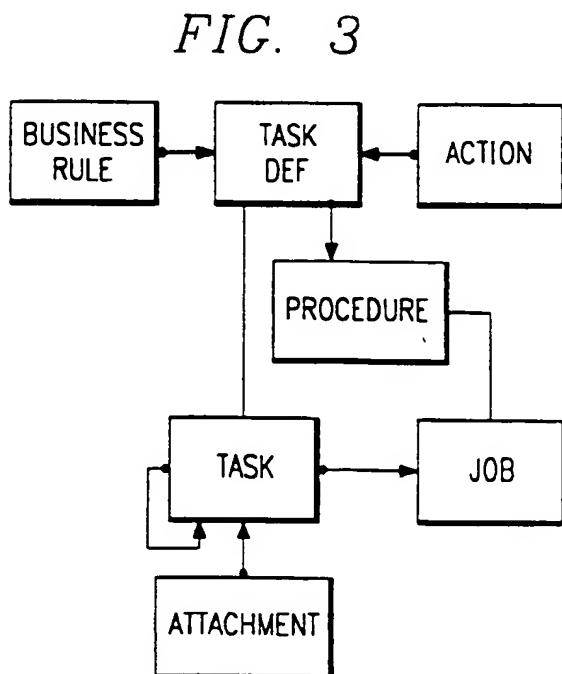
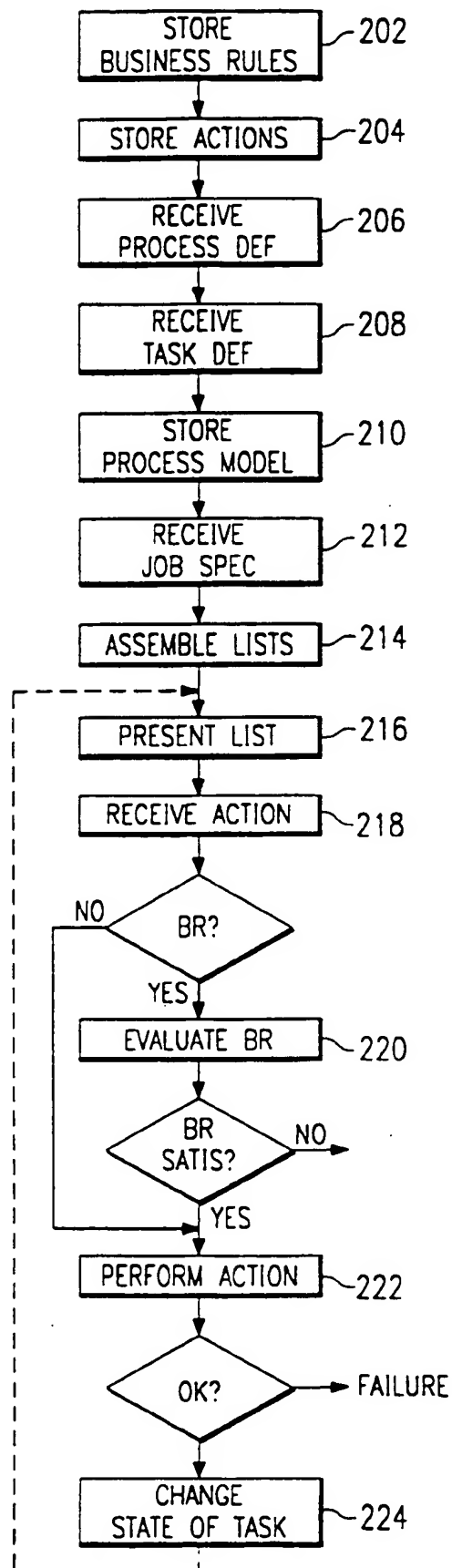


FIG. 3

FIG. 2



2 / 7

FIG. 4

40

PROCESS DEFINITION DIALOG

NAME:

DESCRIPTION:

NAME:	DESCRIPTION:	INVOKE
PROCESS-N OWNER-RELEASE DETAILING-DEPARTMENT-RELEASE ENGINEERING-RELEASE		↑ ↓

FIG. 5

50

TASK DEFINITION DIALOG

NAME:

DESCRIPTION:

ACTION	ACTION HANDLERS	QUORUM	BR HANDLERS	PRIV	NOT
START	INHERIT SET-PROTECTION	1	CHECK-ROLE CHECK-COMPLETION	OR	↑ ↓
COMPLETE	DO				

**HANDLER DECLARATION DIALOG**

NAME:

ARGUMENTS

QUORUM=3  
 DETAILING-DEPARTMENT-RELEASE[OVERRIDE]  
 CHECKER

60

FIG. 6

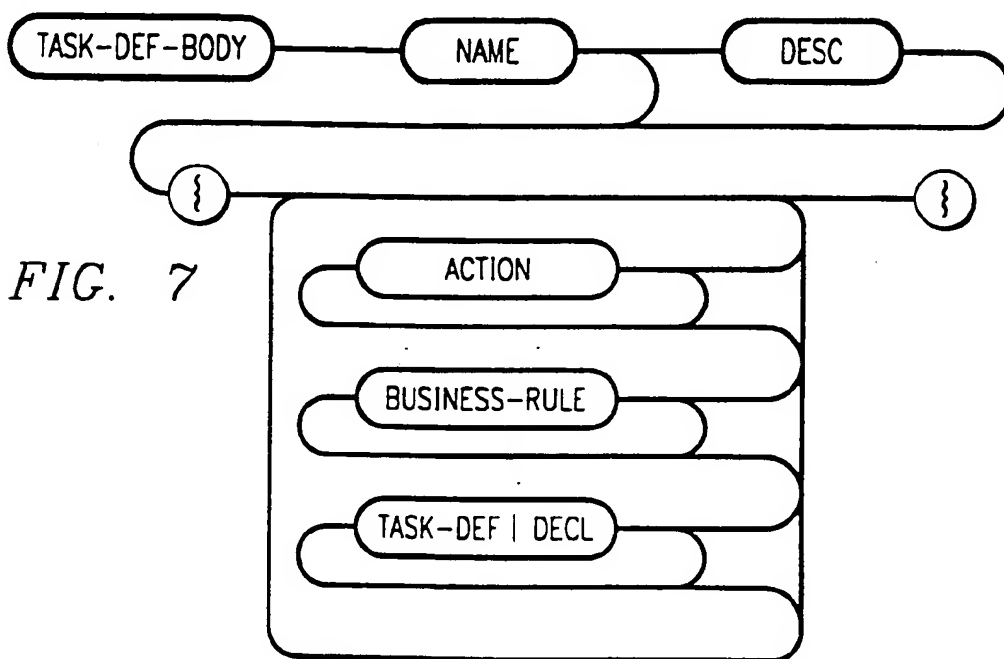


FIG. 7

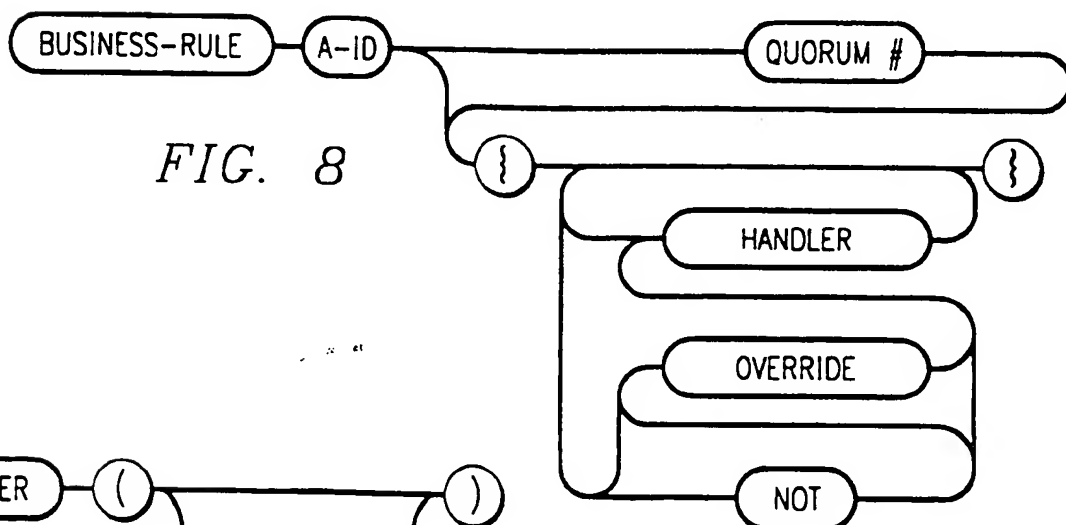


FIG. 8

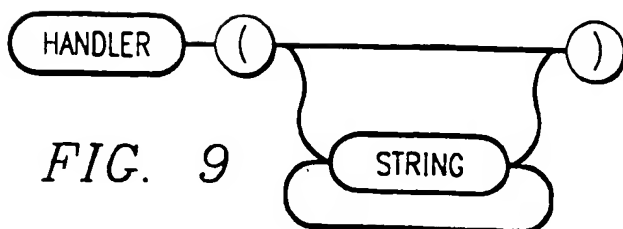


FIG. 9

4 / 7

FIG. 10

100

**JOB DIALOG**

IDENTIFICATION:  DATE CREATED:

DESCRIPTION:  CREATED BY:

TASK NAME	RESPONSIBILITY	STATUS
CHANGE PROCESS	JOE	STARTED
VALIDATE CHANGE REQUEST	MARY-ANN	PENDING
ASSIGN TEAM MEMBERS	JOHN	PENDING
EXECUTE CHANGE	MARK	PENDING

FIG. 11

110

JOB IDENTIFICATION:  STATUS:

TASK DESCRIPTION:

**SUBJECT:**

OBJECT ID	OBJECT TYPE	RELATION
DRW-25-A	UGPART	TARGET
NOTE-25-A	TEXT	REFERENCE
MAT-40S	FORM	STATUS
ECO-25-A	FORM	TARGET

STATUS

5 / 7

FIG. 12

120

IN BOX			
JOB IDENTIFICATION	TASK NAME	OWNER	STATUS
ECO-125-A	VALIDATE CHANGE REQUEST	JOE	PENDING
ECO-127-A	VALIDATE CHANGE REQUEST	JIM	PENDING

FIG. 13

130

CHANGE REQUEST:

RESPONSIBLE PARTY:

NOTE:

FIG. 14

140

IN BOX			
JOB IDENTIFICATION	TASK NAME	OWNER	STATUS
ECR-125-A	VALIDATE CHANGE REQUEST	MARY-ANN	PENDING
ECO-127-A	VALIDATE CHANGE REQUEST	JIM	PENDING

6 / 7

FIG. 15

150

JOB IDENTIFICATION:  STATUS:

TASK DESCRIPTION:

SUBJECT:

OBJECT ID	OWNER	OBJECT TYPE	RELATION	
ECR-125-A	MARY-ANN	CHANGE REQUEST FORM	REFERENCE	↑ ↓

STATUS

151

FIG. 16

160

JOB IDENTIFICATION:  STATUS:

TASK DESCRIPTION:

SUBJECT:

OBJECT ID	OWNER	OBJECT TYPE	RELATION	
ECR-125-A	MARY-ANN	CHANGE REQUEST FORM	REFERENCE	↑
ECO-125-A	MIKE	ECO FORM	REFERENCE	↓
L-123-E	JOE	UGPART	TARGET	

STATUS

7/7

170

FIG. 17

OBJECT NAME	OBJECT TYPE	OWNER	RESPONSIBLE	STATUS
-CHANGE PROCEDURE REQUEST ECO-123	TASK CHANGE REQUEST FORM ENGINEING CHANGE ORDER	MARY-ANN MARY-ANN BOB	BOB	STARTED
TARGET L-123-E	FOLDER UGPART	BOB JOE		RELEASED
CHANGE REQUEST VALIDATE REQUEST CHANGE DRAWING CHECK CHANGE RELEASE CHANGE	TASK TASK TASK TASK TASK	MARY-ANN MARY-ANN MARY-ANN MARY-ANN MARY-ANN	MARY-ANN BOB LEROY MARY-ANN MARY-ANN	COMPLETED COMPLETED PENDING UNASSIGNED UNASSIGNED

## INTERNATIONAL SEARCH REPORT

Intern. Application No  
PCT/US 94/06688

A. CLASSIFICATION OF SUBJECT MATTER  
IPC 5 G06F15/21

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)  
IPC 5 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP,A,0 510 908 (IBM) 28 October 1992  see column 2, line 1 - column 3, line 17 see column 3, line 45 - column 7, line 3 ---	1,2,4,7, 8,10,11, 14,16
A	COMPUTER COMMUNICATIONS, vol.15, no.8, October 1992, LONDON, GB pages 477 - 488, XP000296989 P. HENNESSY ET AL 'Distributed Work Management: Activity Coordination within the EuroCoOp Project' --- -/--	1,9,13, 15,17

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

## \* Special categories of cited documents :

- \*A\* document defining the general state of the art which is not considered to be of particular relevance
- \*E\* earlier document but published on or after the international filing date
- \*L\* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- \*O\* document referring to an oral disclosure, use, exhibition or other means
- \*P\* document published prior to the international filing date but later than the priority date claimed

- \*T\* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- \*X\* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- \*Y\* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- \*&\* document member of the same patent family

Date of the actual completion of the international search

20 October 1994

Date of mailing of the international search report

04.11.94

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Pottiez, M

# INTERNATIONAL SEARCH REPORT

Internvl Application No  
PCT/US 94/06688

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP,A,0 514 231 (INTELLINOMICS) 19 November 1992 see column 4, line 31 - column 6, line 1 see column 12, line 53 - column 13, line 20 see column 50, line 7 - column 51, line 3 ---	1,3,5,6, 13
A	SIGOIS BULLETIN - CONFERENCE ON ORGANIZATIONAL COMPUTING SYSTEMS, ATLANTA, vol.12, no.2,3, November 1991, USA pages 213 - 224, XP000313812 S.K. SARIN ET AL 'A Process Model and System for Supporting Collaborative Work' see page 214, paragraph 2 - page 215, paragraph 4 -----	1-17

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/US 94/06688

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A-0510908	28-10-92	US-A- 5216592 JP-A- 5089045	01-06-93 09-04-93
EP-A-0514231	19-11-92	NONE	